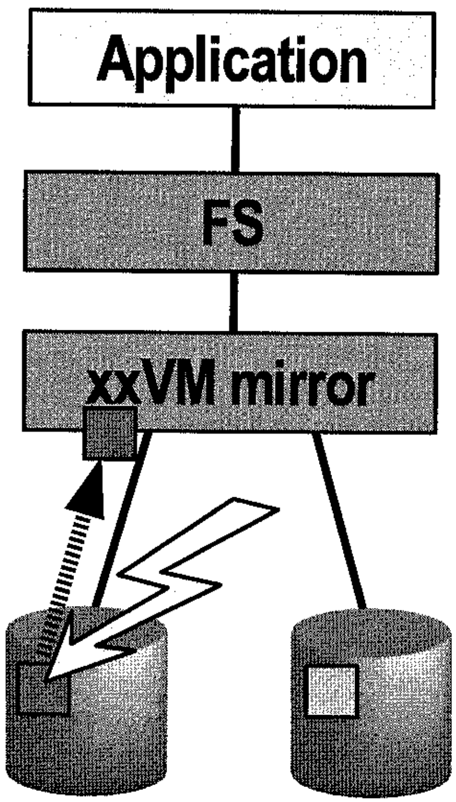
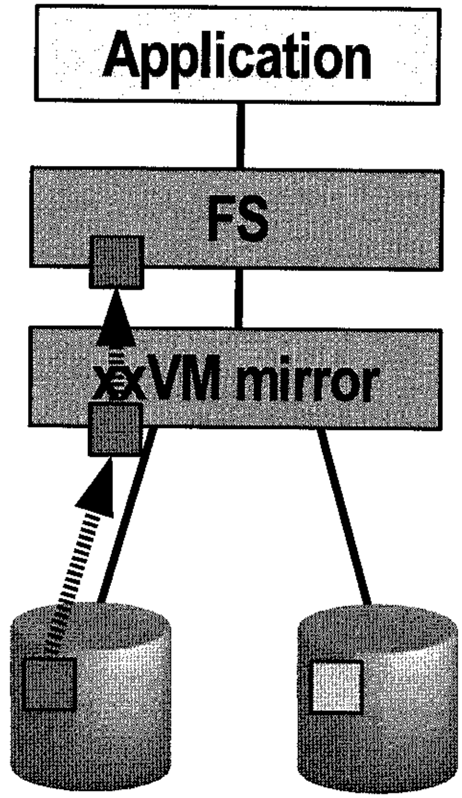
Traditional Mirroring

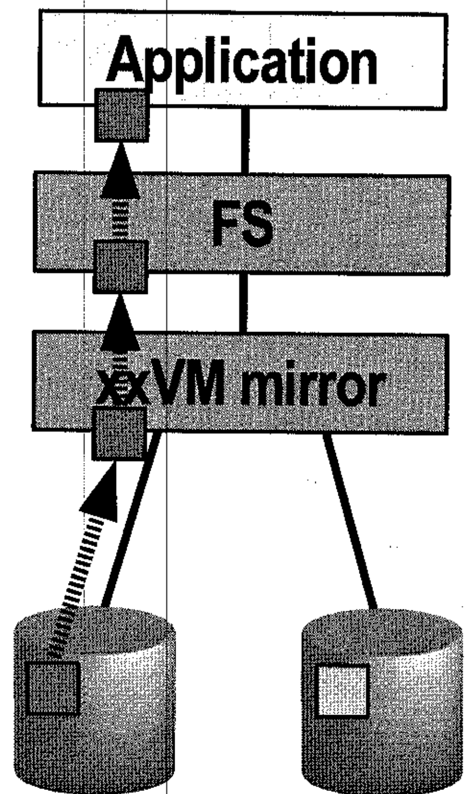
1. Application issues a read. Mirror reads the first disk, which has a corrupt block. It can't tell.



2. Volume manager passes bad block up to filesystem. If it's a metadata block, the filesystem panics. If not...



3. Filesystem returns bad data to the application.

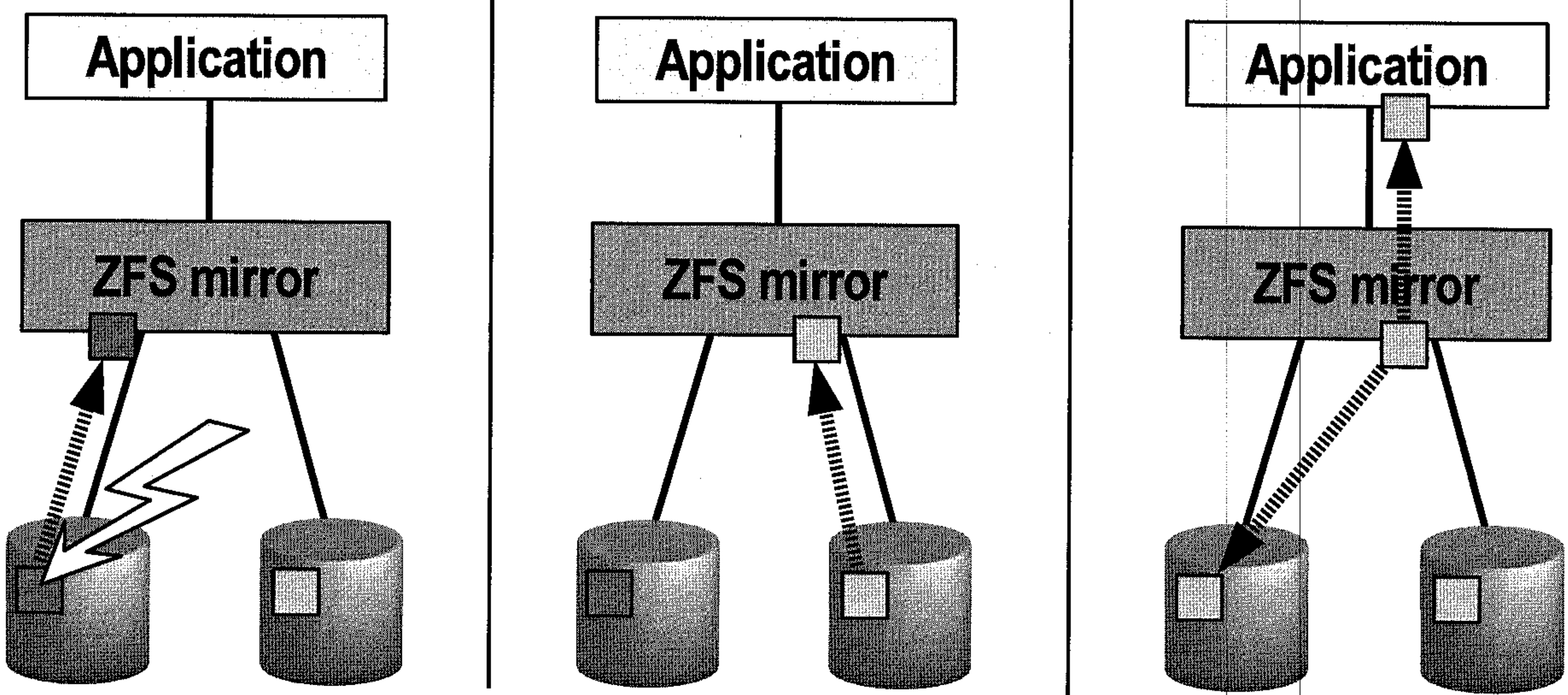


Self-Healing Data in ZFS

1. Application issues a read. ZFS mirror tries the first disk. Checksum reveals that the block is corrupt on disk.

2. ZFS tries the second disk. Checksum indicates that the block is good.

3. ZFS returns good data to the application and repairs the damaged block.

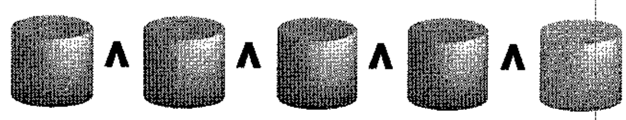


//

Traditional RAID-4 and RAID-5

- **Several data disks plus one parity disk**

 = 0

- **Fatal flaw: partial stripe writes**
  - **Parity update requires read-modify-write (slow)**
    - Read old data and old parity (two synchronous disk reads)
    - Compute new parity = new data ^ old data ^ old parity
    - Write new data and new parity
  - **Suffers from *write hole:***  = garbage
    - Loss of power between data and parity writes will corrupt data
    - Workaround: $$$ NVRAM in hardware (i.e., don't lose power!)
- **Can't detect or correct silent data corruption**

RAID-Z

- **Dynamic stripe width**
  - Variable block size: 512 – 128K
  - Each logical block is its own stripe
- **Both single- and double-parity**
- **All writes are full-stripe writes**
  - Eliminates read-modify-write (it's fast)
  - Eliminates the RAID-5 write hole (no need for NVRAM)
- **Detects and corrects silent data corruption**
  - Checksum-driven combinatorial reconstruction
- **<u>No special hardware – ZFS loves cheap disks</u>**

LBA \ Disk	A	B	C	D	E
0	P_0	D_0	D_2	D_4	D_6
1	P_1	D_1	D_3	D_5	D_7
2	P_0	D_0	D_1	D_2	P_0
3	D_0	D_1	D_2	P_0	D_0
4	P_0	D_0	D_4	D_8	D_{11}
5	P_1	D_1	D_5	D_9	D_{12}
6	P_2	D_2	D_6	D_{10}	D_{13}
7	P_3	D_3	D_7	P_0	D_0
8	D_1	D_2	D_3	X	P_0
9	D_0	D_1	X	P_0	D_0
10	D_3	D_6	D_9	P_1	D_1
11	D_4	D_7	D_{10}	P_2	D_2
12	D_5	D_8	•	•	•

Case 3:07-cv-03026-EDL Document 1-3 Filed 01/08/2007 Page 5 of 10

Traditional Resilvering

- **Creating a new mirror (or RAID stripe):**
  - Copy one disk to the other (or XOR them together) so all copies are self-consistent – even though they're all random garbage!
- **Replacing a failed device:**
  - Whole-disk copy – even if the volume is nearly empty
  - No checksums or validity checks along the way
  - No assurance of progress until 100% complete – your root directory may be the last block copied
- **Recovering from a transient outage:**
  - Dirty region logging – slow, and easily defeated by random writes

Sun microsystems

Smokin' Mirrors

- Top-down resilvering
  - ZFS resilvers the storage pool's block tree from the root down
  - Most important blocks first
  - Every single block copy increases the amount of discoverable data
- Only copy live blocks
  - No time wasted copying free space
  - Zero time to initialize a new mirror or RAID-Z group
- Dirty time logging (for transient outages)
  - ZFS records the transaction group window that the device missed
  - To resilver, ZFS walks the tree and prunes by DTL
  - A five-second outage takes five seconds to repair

Ditto Blocks

- Data replication above and beyond mirror/RAID-Z
  - Each logical block can have up to three physical blocks
    - Different devices whenever possible
    - Different places on the same device otherwise (e.g. laptop drive)
  - All ZFS metadata 2+ copies
  - Settable on a per-file basis for precious user data (snv_61)
- Detects and corrects silent data corruption
  - If the first copy is missing or damaged, try the ditto blocks
  - In a multi-disk pool, ZFS survives any non-consecutive disk failures
  - In a single-disk pool, ZFS survives loss of up to 1/8 of the platter
- <u>ZFS survives failures that send other filesystems to tape</u>

Sun microsystems

Disk Scrubbing

- Finds latent errors while they're still correctable
  - ECC memory scrubbing for disks
- Verifies the integrity of all data
  - Traverses pool metadata to read every copy of every block
    - All mirror copies, all RAID-Z parity, and all ditto blocks
  - Verifies each copy against its 256-bit checksum
  - Self-healing as it goes
- Minimally invasive
  - Low I/O priority ensures that scrubbing doesn't get in the way
  - User-defined scrub rates coming soon
    - Gradually scrub the pool over the course of a month, a quarter, etc.

ZFS Scalability

- Immense capacity (128-bit)
  - Moore's Law: need 65th bit in 10-15 years
  - ZFS capacity: 256 quadrillion ZB (1ZB = 1 billion TB)
  - Exceeds quantum limit of Earth-based storage
    - Seth Lloyd, "Ultimate physical limits to computation." Nature 406, 1047-1054 (2000)
- 100% dynamic metadata
  - No limits on files, directory entries, etc.
  - No wacky knobs (e.g. inodes/cg)
- Concurrent everything
  - Byte-range locking: parallel read/write without violating POSIX
  - Parallel, constant-time directory operations

ZFS Performance

- Copy-on-write design
  - Turns random writes into sequential writes
  - Intrinsically hot-spot-free
- Multiple block sizes
  - Automatically chosen to match workload
- Pipelined I/O
  - Fully scoreboarded 24-stage pipeline with I/O dependency graphs
  - Maximum possible I/O parallelism
  - Priority, deadline scheduling, out-of-order issue, sorting, aggregation
- Dynamic striping across all devices
- Intelligent prefetch